NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USING SOLID MODELING TECHNIQUES TO CONSTRUCT THREE-DIMENSIONAL ICONS FOR A VISUAL SIMULATOR

by

Jane Stolarski Polcrack

September 1991

Thesis Co-Advisors: Michael J. Zyda
David R. Pratt

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
USING SOLID MODELING TECHNIQUES TO CONSTRUCT THREE-DIMENSIONAL ICONS FOR A VISUAL SIMULATOR (U)

12. PERSONAL AUTHOR(S)
Polcrack, Jane Stolarski

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 08/89 TO 09/91	September 1991	64

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	icon,constructive solid geometry, modeling, computer graphics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Realistic three dimensional (3D) models are an essential part of any battle simulator. They contribute greatly to the quality of the scenarios and the decision making training the system can provide. Commercial programs are available to build and modify these models, also known as icons, but they tend to be very expensive and complicated. They also tend to be very specific as to the file format used to store icons. The developers of the Naval Postgraduate School's battle simulator, NPSNET, need a simple, easy to use, and inexpensive system which allows them to quickly build and modify icons stored in Object File Format (OFF).We present the program NPSICON to meet this need and also discuss some of the issues involved in building 3D icons. NPSICON runs on commercially available Silicon Graphics, Inc. IRIS workstations.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Michael J. Zyda	22b. TELEPHONE (Include Area Code): (408) 646-2305; 22c. OFFICE SYMBOL: CS/Zk

Approved for public release; distribution is unlimited

USING SOLID MODELING TECHNIQUES TO CONSTRUCT THREE-DIMENSIONAL ICONS FOR A VISUAL SIMULATOR

by

Jane Stolarski Polcrack
Captain, United States Army
BBA, St. Bonaventure University, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

ABSTRACT

Realistic three dimensional (3D) models are an essential part of any battle simulator. They contribute greatly to the quality of the scenarios and the decision making training the system can provide. Commercial programs are available to build and modify these models, also known as icons, but they tend to be very expensive and complicated. They also tend to be very specific as to the file format used to store icons. The developers of the Naval Postgraduate School's battle simulator, NPSNET, need a simple, easy to use, and inexpensive system which allows them to quickly build and modify icons stored in Object File Format (OFF).We present the program NPSICON to meet this need and also discuss some of the issues involved in building 3D icons. NPSICON runs on commercially available Silicon Graphics, Inc. IRIS workstations.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Mike Zyda for introducing me to the magical world of computer graphics and the possibilities it offers for the future, for being there with words of encouragement whenever the development of NPSICON was not going well, and for clearly focusing the direction of my thesis.

I would like to thank my other thesis advisor Dave Pratt for his help, patience, and words of wisdom, without which there would be no NPSICON.

I would also like to thank Commander Rachel Griffin for teaching me the C programming language. She agreed to teach the class even though it had not been scheduled for that quarter and meant an overload to her already full teaching schedule. Without that prior knowledge of C, my ability to grasp the basics of computer graphics would have been immensely more difficult.

Many thanks are due to Lieutenant Dave King and Lieutenant Commander Rich Prevatt for the use of their user interface development program, the NPS Panel Designer and ToolBox. It made developing NPSICON's user interface much easier and helped give NPSICON an easy-to-use and professional-looking user interface.

Many thanks are also due to Captains Bill Osborne and Randy Mackey who worked with me on my first graphics project, a program called "test_drive" which was a very simplified early version of what eventually became NPSICON. They helped me get started and provided much needed help and encouragement along the way.

Finally, I would like to thank my other half, Erik, who is truly my better half. Without his help, moral support, love, and encouragement this thesis would never have been completed.

I. INTRODUCTION

Battle simulators provide useful training at a fraction of the cost of actual field training, with the added advantage of no physical danger to participants. They also provide the ability to repeat scenarios and analyze results in a pictorial manner. Over the years, battle simulators have become more and more important to the military. In view of the present emphasis on budget-cutting in Congress, the cost advantage provided by training on simulators will become even more important. Part of the success of a simulator lies in its ability to realistically depict a battle scene - the terrain, participants, vehicles, weapons, etc. Three-dimensional representations of objects within simulators are often referred to as three dimensional (3D) icons.

A. FOCUS

Users generally think of three dimensional icons in one of three ways: (1) as a whole indivisible object, (2) as a collection of separate parts, and (3) as a set of indivisible polygons. How a user views a three dimensional object depends on what the user is trying to do. Often the same user wants to view the icon differently at different times.

For instance, if a user has created a new virtual world and wants to place an already existing icon in it, he will want to look at the icon as a single whole object which he may need to rotate, scale, or translate to make it look appropriate in the virtual world. If, on the other hand, the user needs to build a three dimensional icon which does not yet exist, he will probably want to view this new icon as a collection of separate parts. Finally, if the user is trying to minimize the number of polygons being displayed, he might want to view the icon as a collection of individual polygons so he can check and see if some hidden polygons could possibly be removed.

Commercial programs are available to deal with icons on all three of these levels, but they tend to be very expensive and complicated. They also tend to be very specific as to the file format used to store icons. The developers of the Naval Postgraduate School's battle simulator NPSNET [Zyda91] need a simple, easy to use, and inexpensive system that allows them to quickly build and modify icons stored in Object File Format (OFF) [Zyda90]. We present the program NPSICON to meet this need. We also discuss some of the issues involved in building 3D icons.

B. CHAPTER SUMMARY

Chapter II describes some 3D icon modification and generation systems built by previous NPS students. Chapter III provides an overview of NPSICON and its development. Chapter IV discusses some techniques used to build NPSICON, including the file format and interface generation system, the NPS Panel Designer and ToolBox (NPSPD). Chapter V covers some specific implementation topics for NPSICON including the methods used to pick, place, and adjust parts. Chapter VI addresses the capabilities and limitations of NPSICON and provides some suggestions for further research and system enhancement. Chapter VII summarizes the conclusions reached on NPSICON and 3D icon construction in general. Appendix A presents the user's manual for NPSICON.

II. OTHER 3D ICON MODIFICATION/GENERATION SYSTEMS AT NPS

Many commercial computer-aided design (CAD) systems are available. They tend, however, to be very complicated to operate. Since we desire a system which is easy to operate, even for a novice user, we ignore these systems. Instead, we examine two of the existing 3D icon modification and generation systems at NPS: 3DShips and Preview.

A. 3DSHIPS

3DShips is a program developed by Daniel Nagel at the Naval Postgraduate School to allow users to interactively create new icons for ships from different types of parts: armament, hulls, superstructures, and masts [Nage89]. The complexity of the ships built is determined by the sum of the parts used to construct it. 3DShips presents users with a window which shows them top and side views of the ship being built, a coordinates window listing the x, y, and z coordinates with respect to the ship being built where the center of the next part will be placed or the last part is currently located, a short list of instructions, and some icons used to obtain menus of available options or to perform special operations (Figures 2.1 and 2.2).

Selecting the icon for a particular part type such as hulls, presents the user with a pop-up menu containing the parts available to the user. To see what each part looks like the user must first select it from this popup menu. 3DShips contains no preview capability. If, however, the user doesn't like the part selected, he can delete it.

Once the user has selected the part, he can move it around using a dial box which has dials for changing the direction with respect to the x, y, and z axes. It is the user's responsibility to determine when the part is properly attached to the ship. 3DShips has no built in capability to assist the user in determining proper attachment.

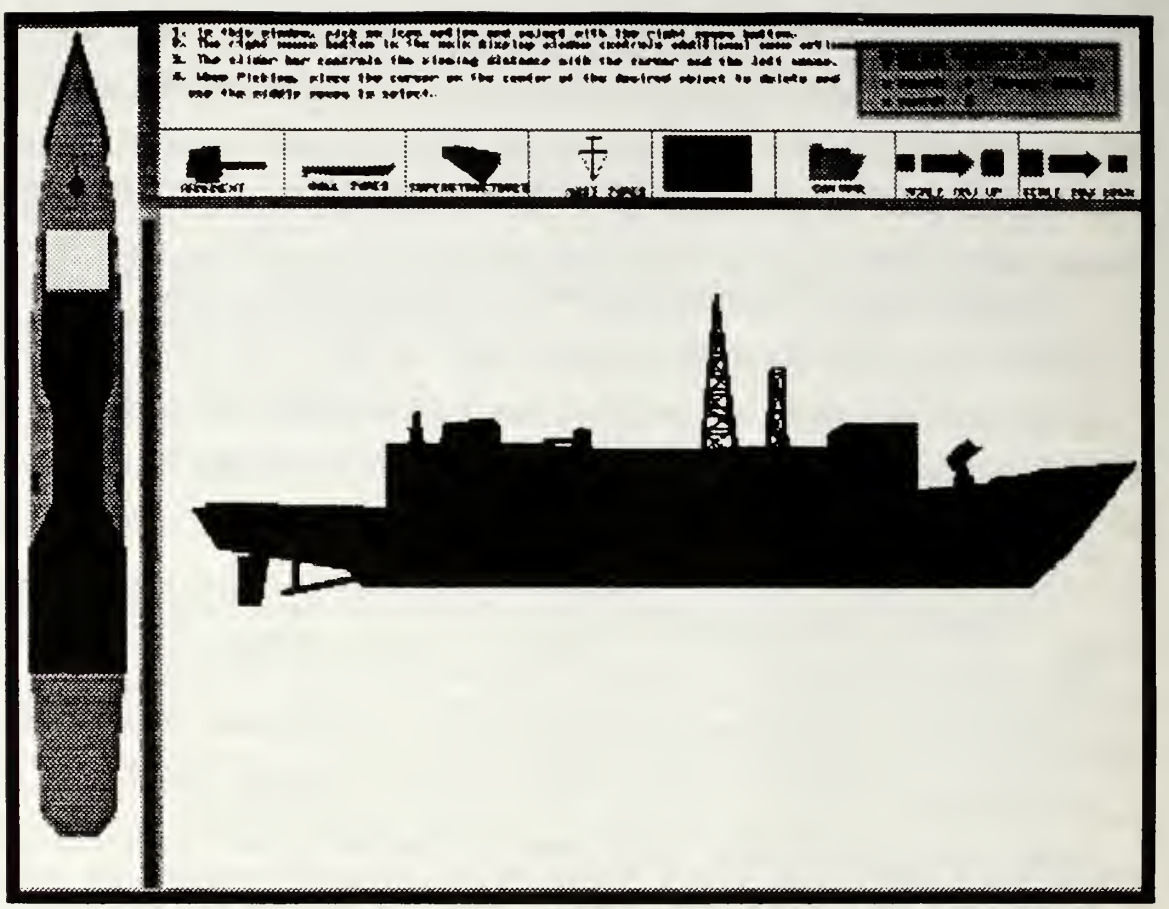

Figure 2.1 - 3DShips' Display

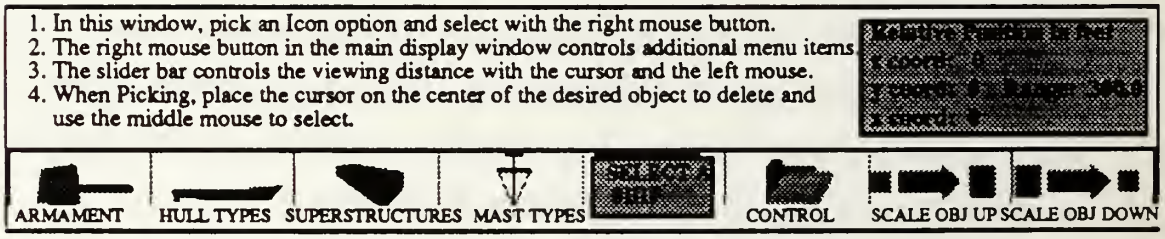

Figure 2.2 - 3DShips' Main Instruction and Icon Window

The user can scale parts up and down by clicking the right mouse button when it is located within one of the icon boxes for scaling. Each click of the mouse button applies a scale factor of 0.9 for scaling down and 1.1 for scaling up.

He can also rotate a part 180 degrees at a time. Rotation of 180 degrees was selected because all objects on a ship typically face the front or back of the ship. Once the user moves on to add the next part, he cannot go back and adjust a previously placed part, other than to delete it.

Newly built ships are saved by clicking the right mouse button over the Save icon. They are saved in sequentially numbered files. The first ship saved is Ship1.new, the second is Ship2.new, etc. The user has no ability to select the name given to his ship, but once saved ships are added to the pop-up menu list under the icon Select Ships. The user can then call each one up and look at it. He can also add additional parts to it. He cannot, however, remove individual parts from previously built ships. The entire ship is considered to be a single unit.

3DShip's user interface is simple. It provides some good ideas on capabilities needed in an icon-building system.

B. PREVIEW

Preview is a tool created by Steven Munson at the Naval Postgraduate School [Muns89]. It is designed to help users manipulate/modify previously created 3D models built using the OFF file format (Figure 2.3). It allows users to scale, translate, and rotate the models and save the changes to a file with the original name suffixed by ".new". As in the 3D Ship program, the user is unable to select the name of the file into which his revised model is saved. Preview also allows users to adjust the resolution level of the displayed polygon.

Prior to loading a model Preview determines the model's origin and the minimum and maximum x, y, and z coordinates and sorts all polygons from largest to smallest. Preview uses this information to determine the appropriate coordinate system scale for the display

window. This helps ensure the system will be able to properly display models of all sizes. This information is also used to adjust the resolution level of the model.

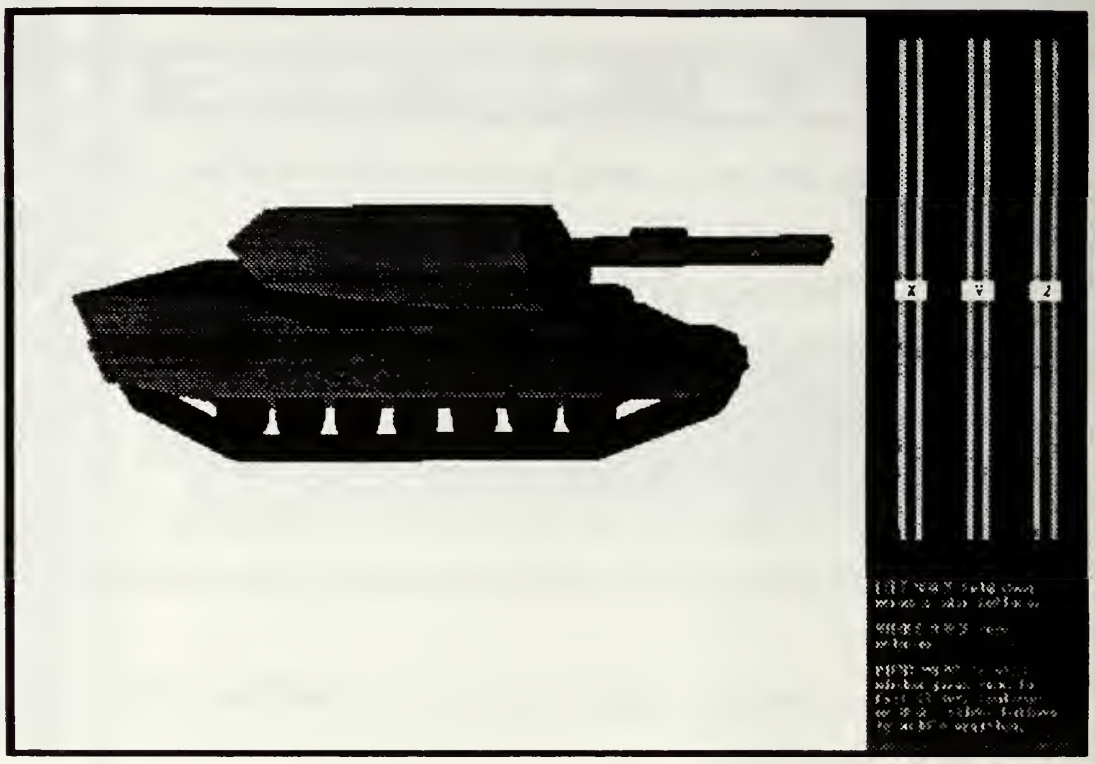

Figure 2.3 - Rotating a Model in Preview

Preview allows the user to view the model as it would normally be displayed or as a "see-thru" wire frame where only the outline of each polygon is drawn. The wire frame version is very helpful when the polygon the user wants is in back of another polygon.

Preview has separate viewing and edit modes. In the viewing mode, the user can rotate and translate the model using sliders. These changes, however, only affect the user's view of the model. The actual model itself remains unchanged. If the user wants to actually change the model, he must switch to the edit mode and start over. In addition to rotation and translation, the edit mode also allows the user to scale the models. All changes to the rotation, translation, and scaling of the object, however, must be done by providing the

precise angles, units of movement, or scale factor. The user cannot work interactively with sliders as he does in the viewing mode.

Preview also allows the user to cycle through the individual polygons within the model and to reverse their normal vectors. Users are allowed to adjust the resolution of their object by selecting which polygons to display. Lower resolution is generally desired for models viewed from a distance. Since Preview has already sorted all polygons from largest to smallest, it is easy for users to identify the small polygons which can not be seen at lower levels of resolution. The elimination of unneeded polygons can help improve the speed of a graphics system, particularly if there are many models to be displayed.

Normal vectors are used to determine the front and back sides of polygons. When normal vectors are computed automatically, it is often difficult for the program to select the proper direction for the normal vector and hence the back side, instead of the front side, of the polygon may be displayed. Preview gives the user an easy way to correct this problem.

While Preview does not allow the user to add or subtract entire parts from an off file, it does allow users to delete individual polygons and provides some important help in managing already created off files. It is also a fairly simple program to use.

III. OVERVIEW OF NPSICON AND ITS USE

A. INITIAL GOALS

NPSICON is designed to create and maintain the icons needed by NPS's battle simulator, NPSNET. NPSNET uses icons stored in OFF; hence, NPSICON also needs to support OFF. To ensure NPSICON remains a viable tool as OFF changes, it uses standard OFF routines so that any improvements to OFF will automatically be included in NPSICON or will, at least, require a minimum amount of changes to incorporate them into NPSICON.

NPSICON is also designed to improve on the work previously done at NPS on icons in the development of the programs Preview and 3DShips. In particular, Preview's method of loading and storing files needs improvement. Currently, a user can only load one icon into preview at a time. This is done by entering the command **preview** and the name of the file containing the icon. To enter the program to view and or edit another icon, the user must exit the program and repeat this process. Any changes made to the icon while in Preview are stored into a file which is given the original icon's file name appended by the suffix "**.new**". If more than one change is made to the icon and saved during a session with Preview, another "**.new**" is added to the name for each change made. This leads to very long and confusing names when the user finally exits Preview and starts to use these modified versions of his icon. This problem can be eliminated by allowing the user to choose meaningful names for his icons. NPSICON is designed to allow the user to load in and work on as many icons as desired during a single session and to allow the user to choose his own name for any modified versions of his icons.

Preview also allows users to manipulate icons with sliders when in the view mode and by typing in the exact change needed in the edit mode. There are times when users want to

edit their icons and know the exact amount of the change needed. There are other times, however, when they would like to manipulate the icon interactively to get the desired change. Sliders allow users to do this. One of NPSICON's goals is to provide users with more support in manipulating icons when editing them.

3DShips also provides ideas for NPSICON's development. 3DShips allows users to customize ships from the pieces in the system. Pieces cannot, however, be seen until attached to the ship. NPSICON is designed to correct this problem.

B. OVERVIEW

NPSICON is meant to be a general purpose tool for editing and viewing existing icons and creating totally new icons. It has both viewing and editing modes. In the viewing mode, users can rotate, scale and translate an icon using sliders. In the edit mode, users can rotate, scale, and translate and icon using sliders and by typing in the exact amount of change desired. Basic parts are included for bodies, cabs, tires, doors, and windows and can be used to modify existing icons or to build new icons. Support is provided to allow users to examine individual polygons, reverse the normal of any polygon, and to delete any polygon. Users are provided with a directory view to load icons, so an icon can be loaded from any directory provided the user has permission to enter the desired directory. In addition, the user can finish working with one icon and then load in others. Users can choose the directory and file name when saving new or modified icons. More details can be found in the user's manual located in the appendix.

IV. TECHNIQUES USED IN NPSICON

In this chapter, some of the basic techniques used to build NPSICON are discussed. These techniques include lighting, the object file format, and the NPS Panel Designer and ToolBox.

A. LIGHTING

Lighting is important in any graphics system because it affects the appearance of all displayed objects [Hall89: pp. 9-11]. We use light to determine an object's shape and color as well as other details about an object. In the real world, when light strikes an object, some light is absorbed into the object, and some light is reflected off the object. How much depends on the light's intensity and placement as well as the material from which the object is made. To obtain a realistic rendition of an object, the proper lighting model must be used. It should closely match the absorption and reflection of light found in the real world. In a real-time system, it cannot be so complex that it prevents the graphics system from running in real time. The lighting model for NPSICON was chosen with these same thoughts in mind.

NPSICON uses white light because it allows the true color of icons to be seen. In the edit mode, a single light is centered on the origin and placed at infinity in the direction from which the user is viewing the object. Ambient light (reflected direct light) is set at 40% to enhance the display of polygons used as windshields in vehicles. In the view mode in which the user's view point, and not the icon, is moved, four lights are used. They are placed to the front, top, and both sides of the icon. To prevent the icon from being over lit the ambience of the lights placed on the top and side is cut to 20% and the color reduced to half intensity. Gouraud shading is used throughout to give surfaces a smooth appearance.

B. OBJECT FILE FORMAT (OFF)

NPSICON is designed to create and work on icons stored in Object File Format (OFF). OFF is a system to store three dimensional objects developed by Steven Munson and Professor Michael Zyda at the Naval Postgraduate School [Zyda90].

The OFF system is based on the use of operation codes (opcodes). Opcodes define the properties of 3D icons such as the polygons and lines in them, the materials from which they are constructed, and the lighting used to light them. Opcodes do not exist in the OFF system, as in most file format systems, for 3D primitives such as cubes, spheres, cylinders, etc. There is one opcode for each property or characteristic an icon has. So, for example, to render a red cube in OFF, one would need, at a minimum, six polygon opcodes and one red material opcode.

All of the opcodes pertaining to an OFF object are stored in ASCII files. This ensures that the files can be read and edited by a user using any ASCII text editor.

In addition to the file format, the OFF system also includes a set of standard routines which allow users to display and manipulate OFF objects. A key goal of NPSICON is to make full use of these pre-existing routines. This will allow NPSICON to easily take advantage of any improvements made to or capabilities added to the OFF system.

C. THE NPS PANEL DESIGNER AND TOOLBOX

NPSICON's entire user interface was developed using the NPS Panel Designer and ToolBox (NPSPD), a tool developed by David King and Richard Prevatt to allow users to quickly generate graphical interfaces for their programs [King 90]. NPSPD is based on the creation of windows known as panels to which the programmer adds the actuators needed to communicate with his program's user. Actuators consist of various pre-designed controls which are operated by a mouse. Some of the actuators available in NPSPD include buttons, sliders, dials, directory views, boxes, titles, typeins, and typeouts. Users can customize the actuators chosen by resizing them, changing their colors, adding descriptive labels, and setting initial, minimum, and maximum values.

At any time in the construction process, the user can save any individual panel or all panels to an ASCII file called the intermediate file which contains all information on the panels and their actuators. This file can be used to reload the saved panel(s) into NPSPD for further modifications. The user can also edit this file to make any desired changes.

Once the user has finished completing the panels needed for his program, he then generates source code through NPSPD's Code Manager. Using the file name selected by the user (for an example, we will use "User_Panel" as the file name), the Code Manager generates three files: User_Panel.c, User_Panel_fn.c, and User_Panel.h. User_Panel.c contains all the information about the panels and actuators the user has created, including how they are initialized, displayed and controlled. User_Panel_fn.c contains the functions the user writes to react to the input received from the various actuators. It contains the user's application program. Immediately after generation by the Code Manager, User_Panel_fn.c is basically a skeleton of functions to be filled in later by the user. User_Panel.h is the header file for User_Panel.c and User_Panel_fn.c. It contains global definitions and defines the maximum number of panels and actuators used.

The source code generated by the Code Manager sets up a system which will automatically handle the overhead associated with the panels and actuators and then feed the desired information from the actuators to the user's application. Users can either poll the actuators each time through the display loop to detect any changes to actuators or the panel overhead management system can be told to call a certain user function if a certain actuator is activated or changed. Both methods work successfully. The decision as to the method used is made based on the user's application.

NPSICON uses both methods with its actuators. The directory view and all sliders are polled each time through NPSICON's display function to see if any change has occurred. All other actuators (typeins, typeouts, buttons, and listviews) are controlled by having the panel management overhead system call an appropriate function in NPSICON whenever one of these actuators changes. The NPS Panel Designer and ToolBox provides NPSICON with an easy to use and professional looking interface.

In this chapter, we have covered the techniques behind NPSICON's development: its lighting, the OFF system, and the interface generation system used, NPS Panel Designer and ToolBox. All of them play a key role in shaping NPSICON. In the next section, we discuss the actual implementation of NPSICON.

V. IMPLEMENTATION SPECIFICS

In this chapter, we discuss some of the implementation details involved in building NPSICON. The specific details discussed are picking, ray tracing, how parts are attached together, how files are loaded into NPSICON, how icons are saved, and the rotation, scaling, and translation of icons and their individual parts.

A. PICKING VS. RAY TRACING

When building an icon, one of the key concerns of a user is how to tell the system which part the user wants and where he wants it attached. In NPSICON, users pick the type of part desired from a menu. This causes all the parts available within this category to be displayed at the top of the main window. The easiest way for users to pick these parts is to move the cursor over them and then push a mouse button. It is then up to the system to decide which part the cursor is over. Two common methods are available to accomplish this: picking and ray tracing.

Ray Tracing requires decisions to be made as to how many rays to shoot and in which direction [Hanr89: pp. 108-111]. Scanning the whole picture can be time consuming and affect the program's real time performance.

Picking is a built-in method on IRIS workstations to tell which 3D object is near the cursor [Sili90: pp.12-1--12-12]. In order to use picking, one must first define the area which is considered to be near the cursor. This is done by specifying the size of a rectangle in pixels around the cursor, known as the pick size (see Figure 5.1). NPSICON uses a 16 by 16 pixel square as its picksize. One then designates a buffer into which the names of all objects within this rectangle around the cursor will be recorded. Next, the picture is redrawn after turning on pick mode and giving names to each object in the picture.The picture drawn

while pick mode is on is not displayed on the screen. It is only used for the purpose of determining the objects near the cursor. Any named objects which fall within the rectangle around the cursor are recorded in the pick buffer. In NPSICON, the named objects are the parts displayed at the top of the window and the polygons of the icon currently being built. In Figure 5.1, Object 2 would be recorded in the pick buffer as a hit. By reading the hits recorded in the pick buffer, NPSICON is able to determine the next part the user wants to add to his icon and also the place on the icon being built at which he wants to attach the new part.

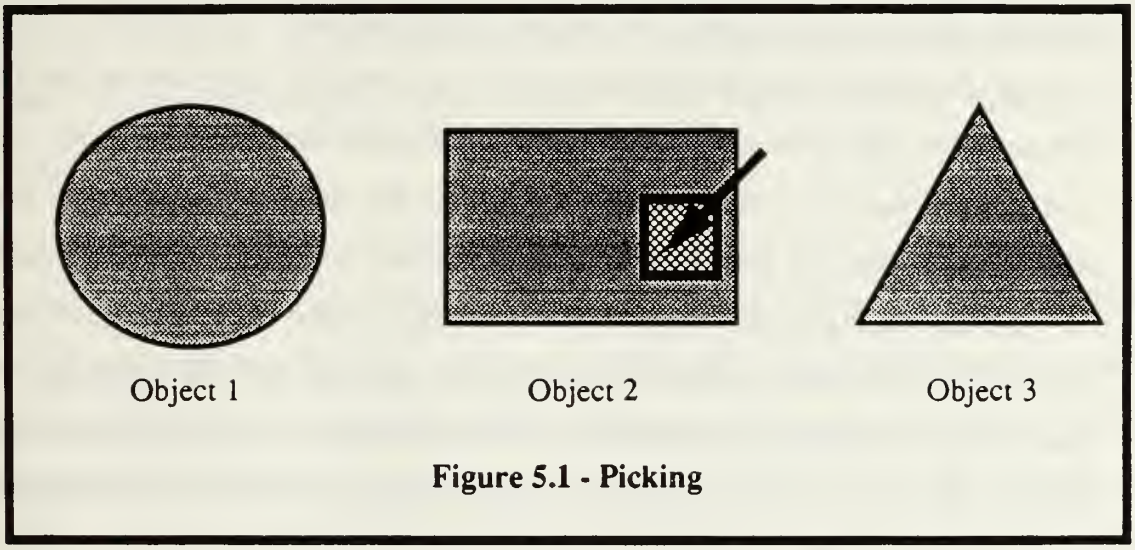

Figure 5.1 - Picking

B. ATTACHING PARTS

To attach two parts together, NPSICON needs to identify the points of attachment on both parts and then rotate and translate the new part so that it is properly attached to the icon. To simplify things, the point of attachment on the new part is assumed to be the center of the back of the part. When the user finishes initially placing a part and releases the middle-mouse button, the coordinates of the cursor provide the x and y coordinates of the place the new part is to be attached. Using picking, NPSICON finds the polygon to which this new part is to be attached. It then plugs in these x and y coordinates into the equation for the plane containing the polygon to which the part is to be attached to come up with the z coordinate.

The amount of rotation needed is the amount needed to rotate the normal vector for the polygon on the front of the new part so that it is parallel to the normal vector at the point of attachment on the icon. Because of the initial placement the normal vector of the polygon at this point is always a unit vector in the positive z direction. The picking process used to get the z coordinate at the point of attachment on the icon also provides the normal vector for the polygon at this point of attachment. This boils down to rotating a unit vector in the positive z direction to become an arbitrary unit vector. Timothy Meier in his thesis on texturing at the Naval Postgraduate School developed a method to rotate an arbitrary unit vector to make it a unit vector in the positive z direction [Mei87: pp. 30-36]. At the same time, he also created the inverse matrix to rotate a unit vector in the z direction back to the arbitrary unit vector selected. NPSICON applies this inverse matrix to the new part to rotate it so that it is parallel to the icon. It then translates this new part to the desired point of attachment. In this way, the new part is cleanly attached to the icon at the proper point.

The drawback with this method is that it requires the part being attached to be a unit vector in the z-direction. For this reason, when an arbitrary OFF file is brought in to be attached to an existing icon, NPSICON provides no support for snapping the new part to the icon. The user must make all the rotation and translation decisions needed to attach the part to the icon.

C. LOADING FILES

To be useful, an icon building/manipulation system needs the ability to load in existing icons. NPSICON allows users to locate files by searching through the files contained in each directory. NPSICON starts this search process from the directory from which the user entered the program. The next time the user requests to load a file in the same program session, he can start from the directory from which he loaded the last file or he can return to the directory from which he entered the program and search from there. Since OFF files are often stored together in the same directory, these default settings make sense.

D. STORING FILES

When storing files, users need to specify both the path and the file name where they want their icon stored. Because users usually store files in a single directory, the directory from which users enter the program is loaded as the default path name. This means that to save icons, most users will only have to enter the file name.

E. ROTATION

In the edit mode, rotation is accomplished by determining the amount of rotation and the axis around which the rotation is desired and then passing these parameters into the OFF function "rotate_this_object." In view mode the same thing is done, but the parameters are passed into the IRIS graphics library function "rotate" which rotates the viewing matrix instead of the icon.

F. SCALING

In the edit mode, scaling via input from the typeins is accomplished by determining the amount of scaling desired in the x, y, and z directions and then passing these parameters into the OFF function "scale_this_object." Scaling via input from the sliders is done by inputting a scale factor of 1.01 each time the slider is moved up and 0.99 each time the slider is moved down. This allows the user to see smooth scaling as he interactively scales the object up or down. In view mode again the amount of scaling and the direction is determined, but the parameters are passed into the IRIS graphics library function "scale" which scales the viewing matrix instead of the icon.

G. TRANSLATION

In the edit mode, translation is accomplished by determining the amount of translation desired in the x, y, and z directions and then passing these parameters into the OFF function "translate_this_object." In view mode the same thing is done, but the parameters are passed into the IRIS graphics library function "translate" which translates the viewing matrix, instead of the icon.

VI. NPSICON LIMITATIONS AND FUTURE DIRECTIONS

NPSCION provides another step forward in the ability to build and manipulate icons. It expands the previous work done in the development of Preview and 3DShips. It provides greater flexibility in dealing with OFF files, but still is short of solving all of a user's needs when dealing with icons. This chapter discusses NPSICON's limitations and some suggestions for future enhancements to it.

A. LIMITATIONS

1. Orientation of Parts to be Added to an Icon

NPSICON only has the ability to automatically attach a part with a normal vector in the positive z direction to an icon.The user is responsible for making all translation and rotation decisions to attach a part which does not contain such a normal vector. To remove this restriction and gain the ability to automatically attach any part to an icon, NPSICON needs an algorithm to rotate an arbitrary normal vector to make it parallel to a desired normal vector. Once such an algorithm is available, the ability to snap any two parts together can be easily added to NPSICON.

2. Ability to Rotate, Scale, and Translate Any Part in an Icon at Any Time

Users need the ability to rotate, scale, and translate all of the parts in an icon at any time. NPSICON only allows users to do this when a part is first added to an icon. Once a subsequent part is added, a user can only adjust a previously added part by deleting it and starting over. This overly complicates the process of adjusting an icon's parts and needs to be remedied in any future version of NPSICON.

3. Stepping Through Polygons by Part or Entire Icon

NPSICON allows users to step through the entire icon's polygons. It is not possible to ask to see just the polygons belonging to one of the icon's parts. Such a capability would help speed up the time needed for users to locate desired polygons.

4. Changing the Color of Individual Polygons

NPSICON provides no capability to change an icon's color. Since color is a key characteristic by which icon's are identified, to be a truly useful tool, NPSICON would have to incorporate the ability to change the color of entire icons, icon parts, and individual polygons.

5. Including Color and Materials When Creating Sub-Icons

Sub-icons are built from individual polygons taken from the currently displayed icon. NPSICON copies all of the data about the desired polygon, except the color or material from which it is made. In most cases, users want the polygon being added to the sub-icon to be the same color or to be made from the same material as it was in the currently displayed icon. NPSICON should not force users to add the color and material definitions. The built-in default for the sub-icon should be the same colors and materials used in the icon.

6. Adding Individual Polygons

NPSICON does not allow users to add individual polygons. There are times when a piece needed for an icon does not exist and cannot be created by scaling an existing piece. In these cases, a user should be able to create a new polygon by indicating where on the screen he would like its vertices placed.

7. Decaling

When two or more objects are nearly co-planar, "tearing," a problem due to z-buffer precision, often occurs [Akel90: pp. 31-37]. In such a case, the graphics system has trouble deciding which polygon is the top one and the objects often have a shimmering

appearance. The user, however, wants to achieve the result of the top object being decaled over the bottom one. This result can be achieved with a Painter's Algorithm. This algorithm draws the objects farthest to nearest and adds some special z-buffer commands. The OFF system already has the capability to draw co-planar objects using this algorithm. Using it, however, requires that the underlying objects and the decaled objects be identified. NPSICON is not presently able to do this, although it could be easily added by modifying the section used to identify the polygon to which a user wants to attach a new part.

8. Problems with the Window Manager

If window boundaries are crossed too quickly, actuators do not function until the boundary is recrossed again slowly. In addition to being annoying to users, this slows down the process of creating and updating icons.

B. FUTURE DIRECTIONS

1. Efficiency Considerations

Much of the inefficiency of NPSICON stems from the inability of the window manager to accurately track the cursor as it moves among the multiple windows used in NPSICON, causing users to often have to recross window boundaries to get actuators to function. If the problem cannot be fixed, NPSICON may need to be written using an interface generation system such as X Windows which does not have this window boundary problem.

2. NPSICON Design Considerations

a. Appearance of Windows

Many of the windows used within NPSICON would have looked better without the borders. Eliminating the borders, however, totally destroys the window managers ability to recognize that the cursor has entered that window. If the problem with the window manager can be fixed or if another interface generation system is used, the

appearance of NPSICON will be greatly enhanced by the elimination of the borders and titles on many of its windows.

b. Color and Material

NPSICON needs to provide users with the ability to choose the materials and colors used to render icons and sub-icons. The user should be able to use both existing colors and materials and to interactively create totally new colors and materials.

c. Attaching Parts

To be a useful system, NPSICON has to provide the ability to snap arbitrary parts together. The current restrictions on this procedure discussed above in Part A.1 are unacceptable.

3. Portability Considerations

NPSICON is designed to run on IRIS workstations and to manipulate OFF icons. The ideas presented on how icons can be created and manipulated, however, are applicable to any graphics hardware and file storage system.

VII. CONCLUSIONS

The development of NPSICON has provided some insight into the capabilities and functions needed in a system to create and manipulate 3D icons. First of all, there are three levels to be addressed: the icon as a single indivisible object, the icon as a collection of parts, and the icon as a set of individual polygons. Each level has its own set of individual needs.

When dealing with an icon as a complete object, users need the ability to rotate, scale, and translate the icon by both exact amounts and interactively using sliders. They also need to be able to perform rotations, scalings, and translations on the viewing matrix, instead of the icon. NPSICON provides all of these functions.

When dealing with an icon as a collection of parts, users need the ability to add new parts and to adjust and resize old parts. NPSICON allows users to add new parts from the system and also new parts supplied by the user from OFF files. While NPSICON can cleanly snap one of its system parts to an icon, it cannot snap a user supplied part to an icon. The user is left to make the necessary rotations and translations to make this happen. Clearly, this is an area for future work. NPSICON allows users to scale, rotate, and translate the part they are currently adding to an icon. Once the part is added, however, NPSICON does not allow them to go back and adjust it, other than to delete it. This is also an area which needs further work.

When dealing with an icon as a set of individual polygons, the user needs the ability to view, add, and delete polygons, as well as the ability to reverse the normal vectors of polygons. NPSICON allows the user to view and delete polygons, as well as reversing their normal vectors. It also allows users to select polygons from an icon and copy them into a

sub-icon. NPSICON does not, however, allow users to create new polygons by specifying the vertices. It also has no capability to specify the color or material of individual polygons.

NPSICON shows that icons can be easily created and manipulated interactively. It provides another step toward a complete system to manage the OFF icons used within the Naval Postgraduate School's battle simulator, NPSNET.

APPENDIX

NPSICON USER'S MANUAL

A. OVERVIEW

NPSICON is a program to allow users to view, manipulate, build, and modify 3D icons or objects stored in the Object File Format (OFF). Unless the user specifically selects the view mode, everything in NPSICON is done in the edit mode. This means that all rotations, scalings, and translations alter the actual model. In the view mode, however, any rotations, scalings, or translations done will have no effect on the icon itself. They only affect how the model looks to the user. In all but the view mode the icon changes because NPSICON is designed to be a "what you see is what you get" system.

B. HOW TO USE NPSICON

NPSICON can be run from any directory by ensuring the user's login shell contains a path to the directory containing NPSICON and by then typing **npsicon**. The directory from which the program is entered becomes the default directory for storing and loading files. Hence, searches for files will start from this directory, and any objects stored by the user will be placed in this directory, unless the user specifies otherwise.

C. INTERACTION WITH NPSICON

Users interact with NPSICON via the mouse and keyboard.

1. Mouse

The mouse, an input device with three buttons which slides on a pad, is the key input device to NPSICON. Its buttons are designated from left to right as left-mouse, middle-mouse, and right-mouse.

a. Left- mouse

The left-mouse is used to control all actuators. These actuators include static menus, buttons, typeins, and sliders and are discussed in Section D of the Appendix.

b. Middle-mouse

The middle-mouse is used to select and initially position parts to be added to the icon being built. Pressing down without releasing the middle mouse button when the cursor is over the desired part allows the user to drag the part to the desired position. Releasing the middle-mouse causes the system to go into its attach part mode.

c. Right-mouse

The right-mouse is used to allow users to make selections from NPSICON's main function menu (Figure A.1). Depressing the right mouse button causes this popup menu to appear. By moving the mouse while still depressing the right-mouse, the user can move through the menu to make his selection. Releasing the right-mouse when the appropriate selection is highlighted causes it to be selected.

2. Keyboard

The keyboard is used by NPSICON to allow the user to enter specific amounts of rotation, scaling, or translation desired. These amounts are entered into typeins (discussed in section D of the Appendix).

D. HOW TO USE NPSICON'S ACTUATORS

NPSICON utilizes a variety of actuators. These include static menus, buttons, sliders, and typeins. Details concerning the implementation of these actuators can be found in [King90].

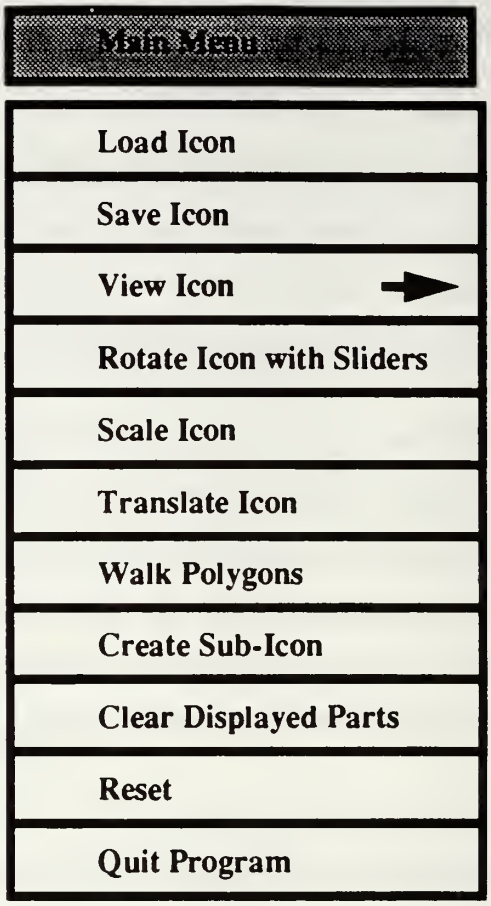

Figure A.1 - NPSICON's Popup Menu

1. Static menus

Static menus are used to provide users with a choice of things. A menu choice is made by moving the cursor via the mouse over the desired menu choice and pressing the left-mouse.

2. Buttons

Buttons are either square or round and are used in NPSICON to turn an option on or off. They are controlled by moving the cursor via the mouse onto the button and then pressing the left-mouse signaling selection of the button.

3. Sliders

Sliders allow users to obtain continuous values between a specified maximum and minimum value. Sliders are controlled by moving the cursor via the mouse onto the blue bar in the slider and then depressing the left-mouse and moving the mouse. Releasing the left-mouse stops all slider movement. Finer control (i.e., slower movement) of the slider can be obtained by holding down the left- and right-mouse buttons simultaneously or by pressing the control key and the left-mouse button at the same time when moving the slider. With the exception of the slider to walk the icon's polygons, all sliders have a reset button which resets all sliders in that window to their initial positions.

4. Typeins

Typeins are used to accept input from the keyboard. An inactive typein appears as a blue box. To activate it and hence signal the user's desire to provide input, move the cursor so that it is in the typein. Then press the left-mouse. This will cause the typein box to turn pink, signaling the typein is ready to accept input. The user can then enter his input from the keyboard. The backspace key can be used to remove any mistyped characters. When the desired input has been entered, the user presses the return key.

E. NPSICON's WINDOWS

NPSICON is a window-based program. Hence, the program communicates with the user by opening up different windows which often contain actuators. If, for any reason, an actuator does not respond, it is probably because the window manager does not recognize that the cursor is in the window containing the desired actuator. To solve this problem, the user should move the cursor outside the window containing the desired actuator and then

slowly move the cursor back into the window containing the actuator. As the cursor crosses the window's boundary, the user should see a red circle with a dot in the middle appear momentarily. This signals that the window manager recognizes the cursor's change in windows. The actuators in this window should now function properly.

F. INITIAL DISPLAY

Initially a single window entitled "npsicon - build/modify icons" is visible (Figure A.2). The top half of this window is blank except for lines indicating the positive x- and y-axes. This top half is used to display any icons being viewed or built. The bottom half contains some basic control actuators: the part menu, typeins to control one time rotation, buttons to control continuous rotation, and some special purpose buttons. It also displays the minimum and maximum bounds in the x, y, and z directions for the displayed icon and the cumulative rotation done to it around the x-, y-, and z-axes.

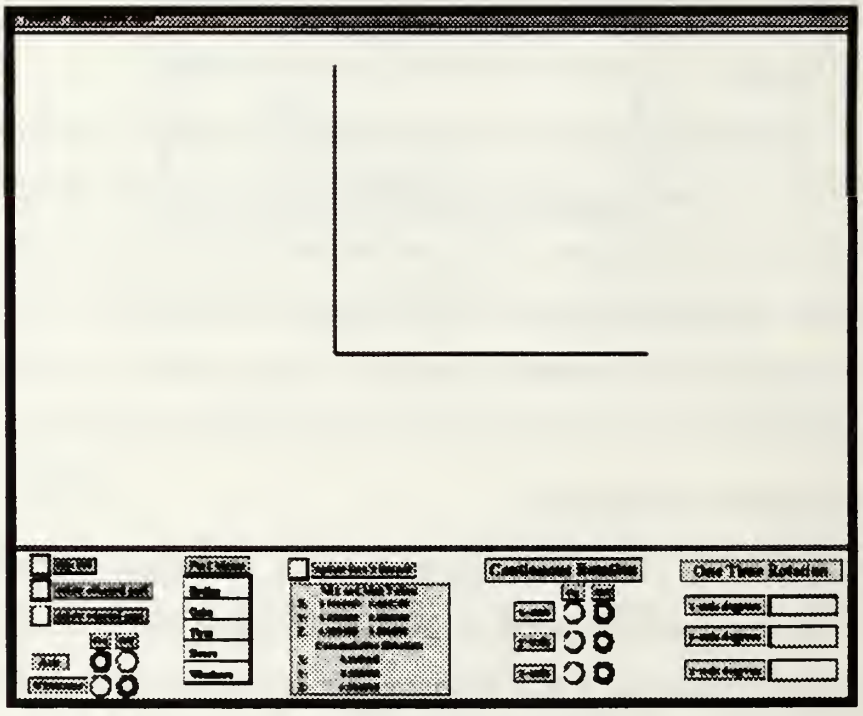

Figure A.2 - NPSICON's Main Window

1. Part Menu

The part menu is used to select parts to be used to build a three dimensional icon. The menu itself lists the categories of parts available: Bodies, Cabs, Tires, Doors, and Windows. Selecting a part category by pressing the left-mouse when the cursor is over the desired part category causes all of the parts available in that category to be displayed at the top of the window.

2. Continuous Rotation Buttons

The continuous rotation buttons allow the user to turn on and off continuous rotation around the x-, y-, and z-axes. Continuous rotation is very useful when the user desires to view his icon from all angles.

3. One-Time Rotation Typeins

The one-time rotation typeins allow the user to input the desired amount of rotation from the keyboard. This is useful when the user wants to rotate an icon an exact amount of rotation around a specific axis.

4. Flip 180 Degrees Button

The "flip 180 degrees" button allows any part, other than the first one, to be rotated 180 degrees. Often when parts are added to the back side of a vehicle the part being added will be rotated an extra 180 degrees. In the case of wheels with hubcaps, this means the part is attached with the hubcap pointing in toward the vehicle instead of away. This function is useful to correct this problem.

5. Delete Attached Part Button

The "delete attached part" button brings up a window which lists all the separate parts in the currently displayed icon [Figure A.3]. Only parts added to the icon in this editing session will be listed. If the current icon was brought in as a complete off file, then it will be considered to be a single part for deletion purposes. In such a case, the user can still delete individual polygons by selecting "Walk Polygons" from NPSICON's popup

menu (Figure A.1). Details on the deletion of individual polygons can be found in Part M of the Appendix.

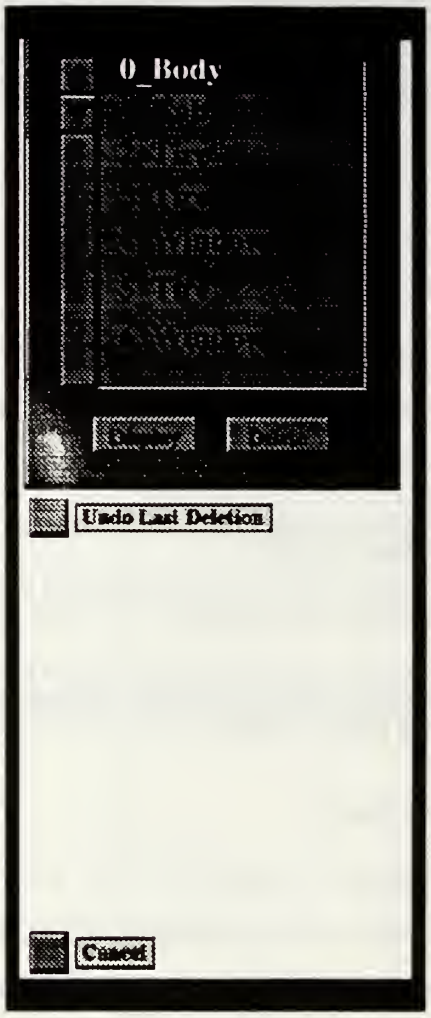

Figure A.3 - Window Used to Delete Parts From an Icon

To select one of the listed parts, the user places the cursor over the name of the desired part and presses the left-mouse. This causes the name of the part to be highlighted in dark blue. He can also use one of the scroll buttons to highlight the name of the part he wishes to consider for deletion. The user then presses the "Display" Button to see the part highlighted in white and outlined in red or the "Delete" Button to delete the part. After a part is deleted, it can be restored by pressing the "Delete Last Part" button. The "Delete Last Part" button only works for the last part which was deleted. Once the user moves on to

delete another part, any previous deletions cannot be undone. Similarly, undoing the last deletion does not allow a user to undo any previous deletions.

6. Delete Selected Part Button

The "delete selected part" button allows a user who has selected a part but not yet attached it to the icon being built to delete the selected part prior to attaching it.

7. Axis On and Off Buttons

To help orient users, NPSICON draws the positive x-, y-, and z-axes in red, green, and blue respectively. The axes which are visible will depend on the user's view point. Initially, the user is presented with a front view and only the x - and y- axes are visible. By following the arrow on the popup menu (Figure A.1) to the right of the entry "View Icon" until a submenu appears and selecting one of its options ("Edit Mode - Front View," "Edit Mode - Side View," and "Edit Mode - Top View"), the user can see all three axes. These axes can be removed from the screen by selecting the Axis Off Button located in the lower left corner of the Main Window. To redisplay the axes, the user follows the same procedure, but selects Axis On Button.

8. Wireframe On and Off Buttons

To display the icon that is currently displayed as a wireframe or skeleton in which only the outline of each polygon is drawn, the user should select the Wireframe On Button located in the lower left corner of the Main Window (Figure A.4). To return to the normal display of the icon, the user should follow the same procedure, except select the Wireframe Off Button.

9. Update Icon's Bounds Button

Whenever an icon is loaded into NPSICON, its minimum and maximum bounds are displayed in the box located in the lower center of the Main Window. If the icon is changed, however, through rotation, scaling, or translation, these displayed bounds will not reflect the changes to the icon's bounds. To obtain the icon's current bounds, the user

presses the "update icon's bounds" button located above the box displaying the icon's bounds.

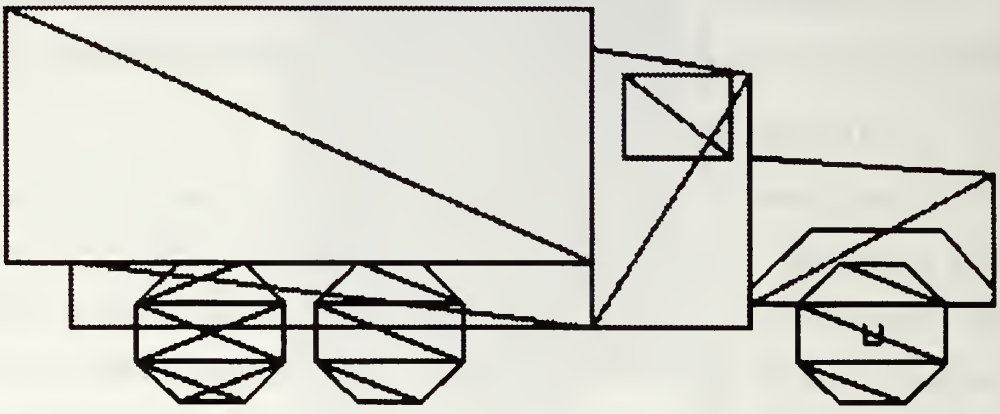

Figure A.4 - Icon Drawn in Wireframe Mode

G. HOW TO LOAD FILES INTO NPSICON

To load files into NPSICON, the user selects the option "Load Icon" from the popup menu (Figure A.1). The user will initially be presented with a directory view listing the files contained in the directory from which the user entered the program (Figure A.5). The user can select any file or directory in this directory by placing the cursor on the desired file or directory name and pressing the left mouse button.This will cause the file or directory name to be highlighted in dark blue. Directory names are followed by a slash "/". To move up one directory from the current directory select the top entry in the directory view "../". The scroll bar and up and down arrows to the left of the directory view can be used to view all of the file names within the current directory.

1. Accept Button

Pressing the accept button causes NPSICON to load the currently selected file into the system or to change directories if a directory is selected. NOTE: (1) When the first

icon is loaded, the user is presented with a front view. The positive x- and y- axes are displayed and the wireframe option is turned off. (2) Subsequent icons will be loaded into the upper left hand corner of NPSICON's main window. The user can then adjust this new icon's position and size using the sliders in the window brought up on the right side of the main window for this purpose. (3) NPSCICON will only load files which end in ".off". If the user presses the accept button and the file currently highlighted does not end in ".off", the system will refuse to load the file and will bring up an error notification window to tell the user of the problem. (4) All OFF files loaded into NPSICON should contain the definitions for any materials used in them. If the material definition is not contained in the OFF file, the user will receive the following error message "retrieve_opcode_by_name: We did not find that name in the name table = <material name>." NPSICON will use its default material, white, in place of the material whose definition was not found.

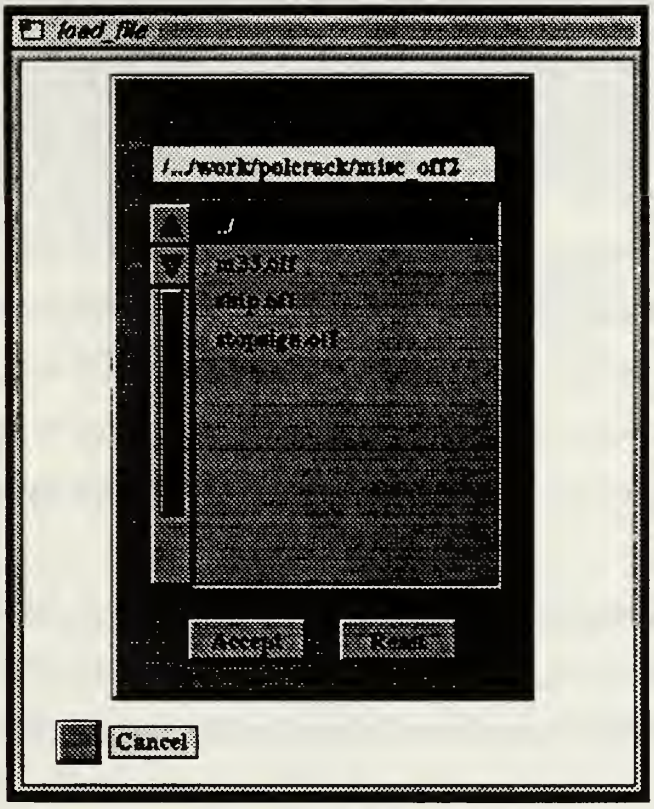

Figure A.5 - Load File Window

2. Reset Button

Pressing the reset button causes the directory view to return to the directory from which the user entered the program.

3. Cancel Button

Pressing the cancel button allows the user to get rid of the directory view without making a selection and to return to the program.

H. HOW TO SAVE NEW OR MODIFIED ICONS

In order to save an icon that has been created or modified, the user must know the directory into which he desires to save the icon and the name he wants to give the icon. The file name into which all icons are stored will end in "**.off**". The system will automatically add the "**.off**" ending if the file name the user selects does not end in "**.off**".

To save an icon in NPSICON, the user selects the option "Save Icon" from the popup menu (Figure A.1). The user will initially be presented with the directory from which he entered the program (Figure A.6). If the user desires to save his new icon in this directory, then he only has to fill in the file name in the file name typein. If the user does not want to save the icon in the default directory, he can change it by typing in the desired path. He must ensure that the path name entered ends with a slash "/". Otherwise, the system will be unable to interpret the path correctly and save the icon where the user desires. Any changes to the directory into which the user wants the icon saved must be done before the user enters the file name. Once the user presses the enter key on the file name typein, the icon will be saved.

If for any reason the system is unable to save the file in the desired directory, the user will be notified via an error notification window. The user will also be notified if the directory already contains a file with the same name the user selected. In this case, the user will be given the option to overwrite the file, overwrite the file but save the original file as a backup, or go back and select another file name. If the user changes his mind about saving the icon, he can return to the program by pressing the cancel button.

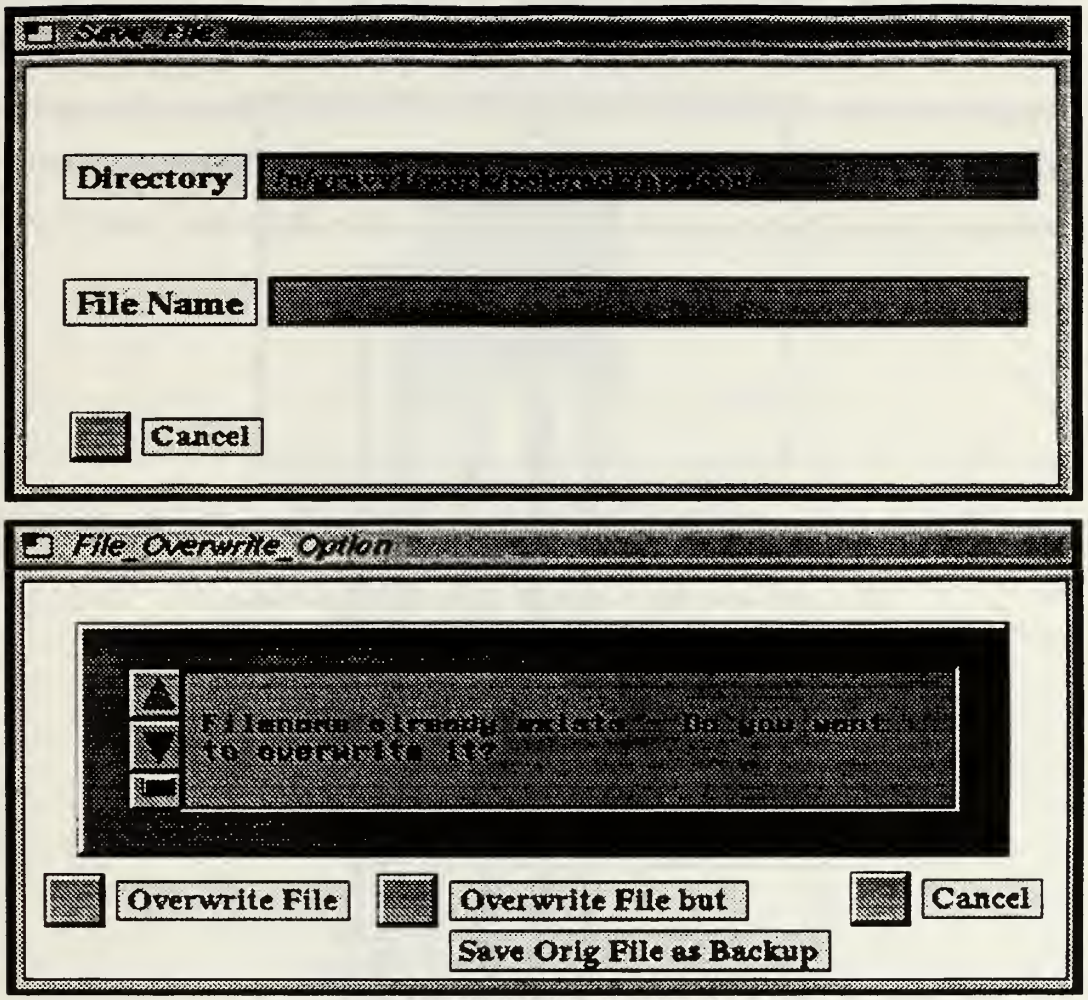

Figure A.6 - Windows Used to Save Files

I. VIEW MODE

The view mode is available to allow users to view icons in different ways without changing the actual model. To enter the view mode the user follows the arrow on the popup menu to the right of the option "View Icon" (Figure A.1) until a submenu appears. The user should then select the option "View Mode" from this submenu. The view mode allows the user to rotate, scale, and translate an icon using sliders (Figure A.7). The user is also given the choice of viewing the icon through an orthographic or perspective viewing projection. NPSICON's default is the perspective viewing projection. It was chosen because it allows

users to visualize translations and scalings in the z direction. All actuators used in the edit mode are disabled when the user enters the view mode; only the actuators in the view mode window will work. To exit the view mode, the user can either select an edit mode view from the submenu under the "View Icon" option on the popup menu (Figure A.1), or the user can push the "Cancel" button located in the lower left corner of the view mode window.

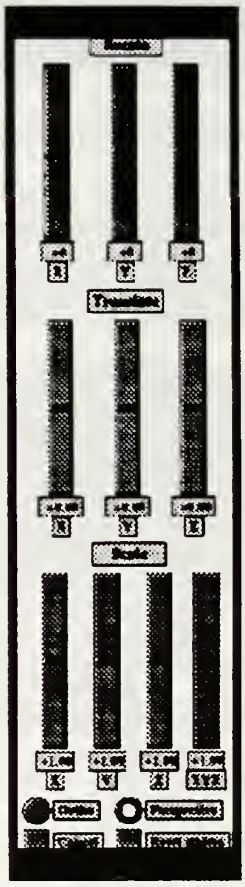

Figure A.7- View Mode Actuators

J. VIEWING THE ICON IN THE EDIT MODE

By following the arrow to the right of "View Icons" on the popup menu (Figure A.1) and selecting "Edit Mode - Front View," "Edit Mode - Side View," or "Edit Mode - Top View" from the submenu which then appears, the user can view his icon from the front, side, or top. The default for the program is the front view. The front view allows the user to view the his icon along the x- and y-axes (Figure A.8). The side view allows him to view

it along the y- and z-axes (Figure A.9). And the top view allows him to view it along the x- and z-axes (Figure A.10).

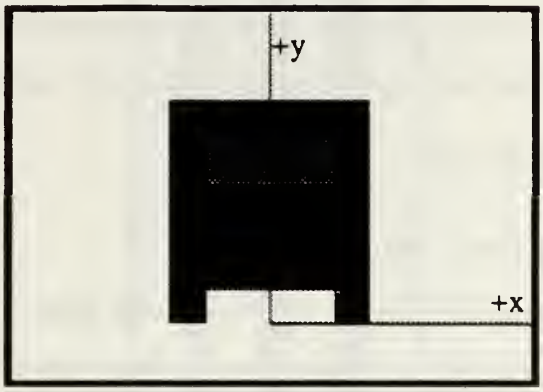

Figure A. 8 - Front View of an Icon

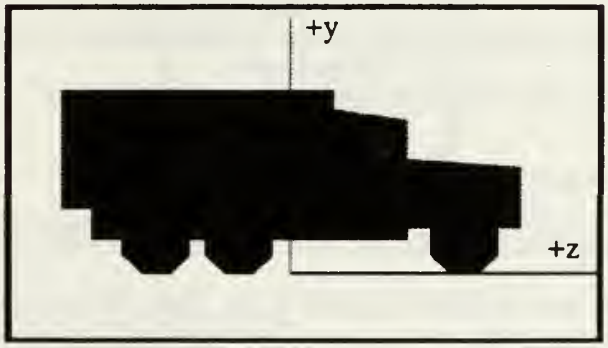

Figure A.9 - Side View of an Icon

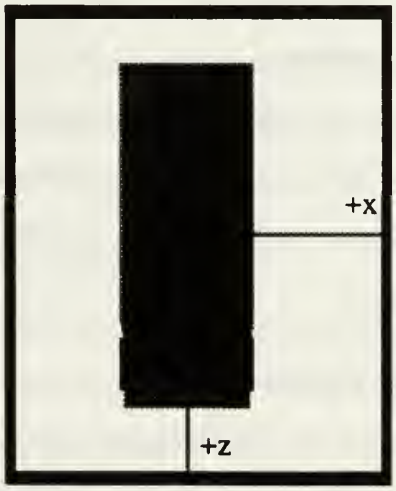

Figure A.10 - Top View of An Icon

K. MANIPULATING ICONS IN THE EDIT MODE

In the edit mode icons can be manipulated by rotation, scaling, and translation. Users always have a choice of using sliders or typeins. For rotation they can also select continuous rotation using the buttons found on the bottom of the main window.

1. ROTATING ICONS

a. Rotating Icons with Sliders

To bring up the rotation sliders, the user selects "Rotate Icon with Sliders" from the popup menu. Initially, all sliders are set to 0.0 (Figure A.11). Using the sliders the user can rotate the icon that is currently displayed 360 degrees in the positive and negative direction around the x-, y-, and z-axes by selecting the appropriate slider. The user can obtain more than 360 degrees rotation by resetting the sliders to zero using the Reset Sliders to Zero Button located in the lower right corner of the window.

b. Rotating Icons with Typeins

The typeins to rotate the icon that is currently displayed are located in the lower middle of the main window (Figure A.2). They allow the user to specify a specific amount of rotation around the x-, y-, and z-axes by typing the desired amount of rotation in the appropriate typein.

c. Continuous Rotation

The buttons to turn on continuous rotation for the icon that is currently displayed are located in the lower right of the main window (Figure A.2). They allow the user to rotate his icon continuously around the x-, y-, and z-axes. Rotation around more than one axis at a time is permitted.

2. TRANSLATING ICONS

To translate the currently displayed icon, the user selects "Translate Icon" from the popup menu. The user can then rotate the icon using sliders or typeins (Figure A.12).

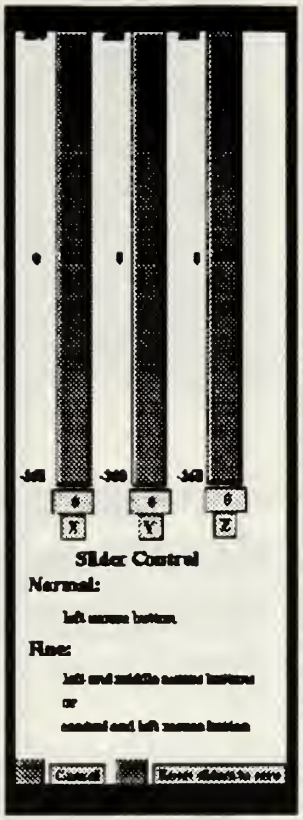

Figure A.11 - Sliders to Rotate an Icon

a. Translating Icons with Sliders

Initially, all sliders are set to 0.0. Using the sliders the user can translate the icon that is currently displayed ten units in the positive and negative x, y, and z directions by selecting the appropriate slider. The user can obtain more than 10 units of translation in one direction by resetting the sliders to zero using the "Reset Sliders to Zero Button" located in the lower right corner of the window.

b. Translating Icons with Typeins

The typeins to translate the icon that is currently displayed allow the user to specify a specific amount of translation in the x, y, and z directions by typing the desired amount of translation in the appropriate typein.

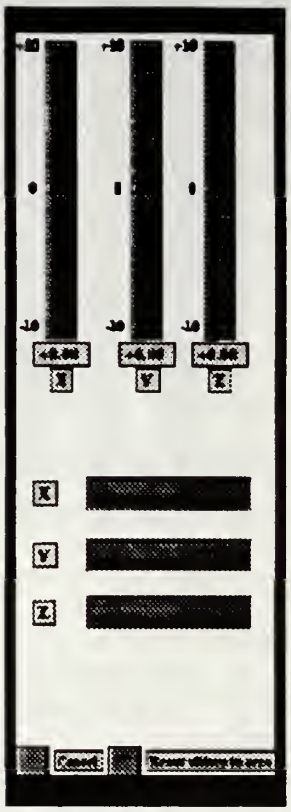

Figure A.12 - Sliders and Typeins to Translate an Icon

3. SCALING ICONS

Users can make the currently displayed icon larger or smaller by selecting "Scale Icon" from the popup menu. The user can then scale the icon using sliders or typeins in the x, y, or z direction individually or in all three directions at once (Figure A.13).

a. Scaling Icons with Sliders

Initially, all sliders are set to 1.0. The user can scale the icon up by moving the appropriate slider up, or he can scale the icon down by moving the appropriate slider down. The user can obtain greater scaling of the icon by resetting the sliders to zero using the "Reset Sliders to Zero Button" located in the lower right corner of the window and again moving the scaling sliders.

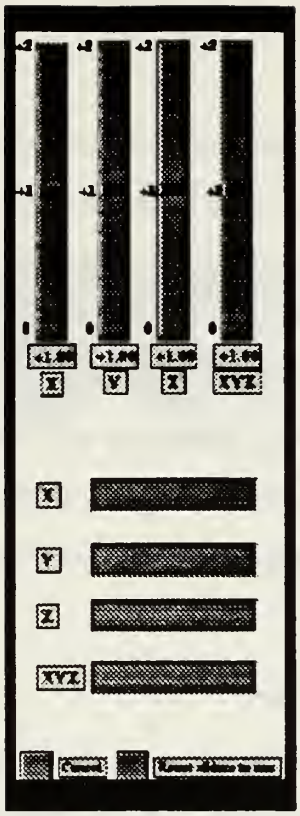

Figure A.13 - Sliders and Typeins to Scale an Icon

b. Scaling Icons with Typeins

The typeins to scale the icon that is currently displayed allow the user to specify a specific amount of scaling in the x, y, and z directions by typing the desired amount of scaling in the appropriate typein. Scaling amounts entered should be positive numbers. A scale amount of 1.0 causes the icon to be scaled to 100% and will leave the icon unchanged. A scale amount of 2.0 will double the size of the icon, and a scale amount of 0.5 will reduce the icons size by one half. Hence, amounts between zero and one decrease the icon's size, and amounts greater than one increase its size.

L. BUILDING ICONS

1. Selecting Parts

The first step in building an icon is to select a part. This can be done by first selecting the category of part desired from the part menu. Selecting a part category from the part menu causes all the parts available in that category to be displayed at the top of the window. Figure A.14 shows how all of the tires in the system are displayed. A user can repeat this process to examine all of the available parts.

Once the desired part is located, the user selects it by pressing and holding down the middle-mouse. This gives the user a copy of the part which he can then drag to the desired position. Releasing the middle-mouse tells the system this is where the part is to be placed.

2. Placement of First Part

The first part a user selects is automatically centered at the origin. This is done to help with the construction process by ensuring all rotations are around the center of the icon. Using the translation option from the popup menu, the user can later move the icon to the desired position.

3. Placement of Subsequent Parts

Placement of subsequent parts is a multi-step process. Upon release of the middle-mouse, the system will highlight in white the polygon it thinks the user wants to attach his part to and show red cross hairs at the point it thinks the user wants to attach the part. The system will then display a window asking the user if this is in fact where the user wants his part placed (Figure A.15). In addition to answering yes or no as to placing the part at the specified location, the user can also cancel the part and try again by pressing the "delete selected part" button found in the lower right corner of the main panel.

If the user selects no, the windows to walk the polygons and to allow the user to move the red placement cross hairs are brought up (Figure A.16). If the polygon to which

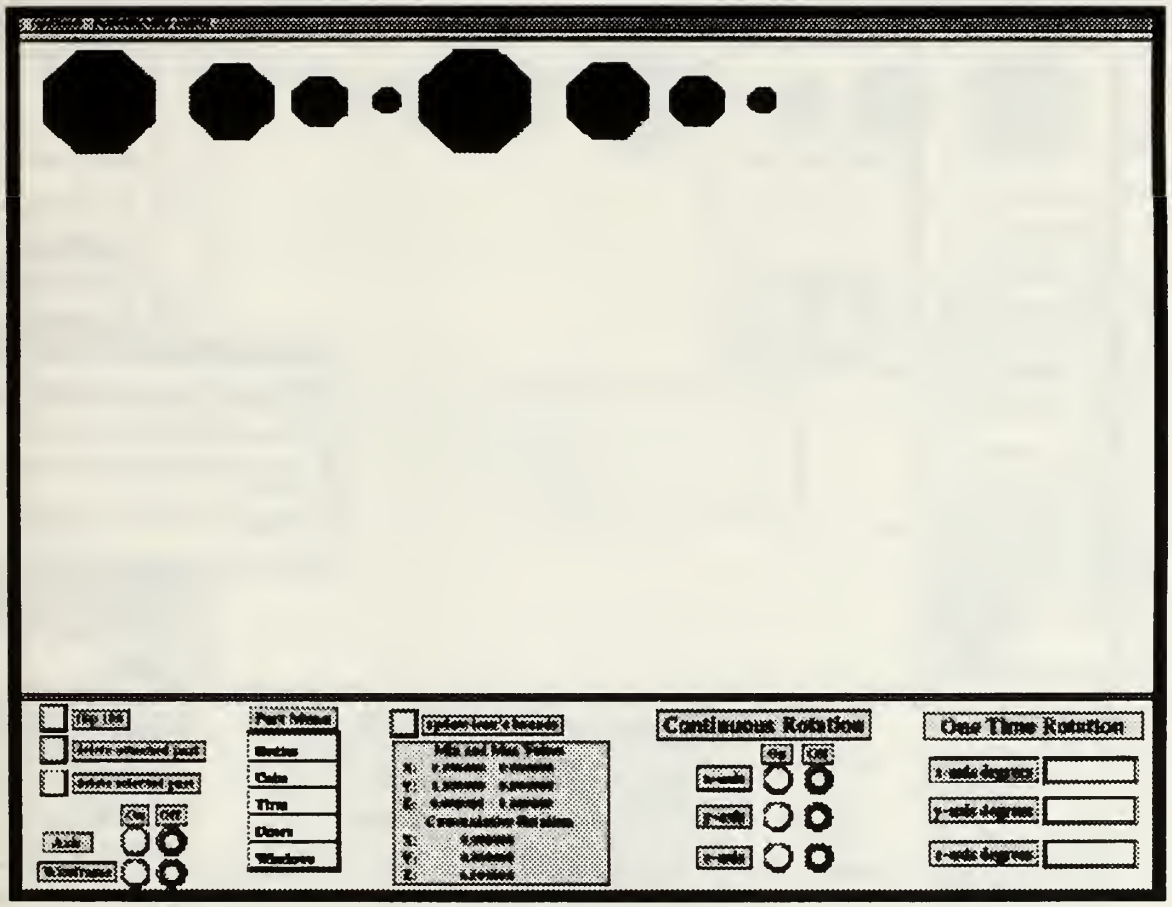

Figure A.14 - Window Displaying All Tires Available Within NPSICON

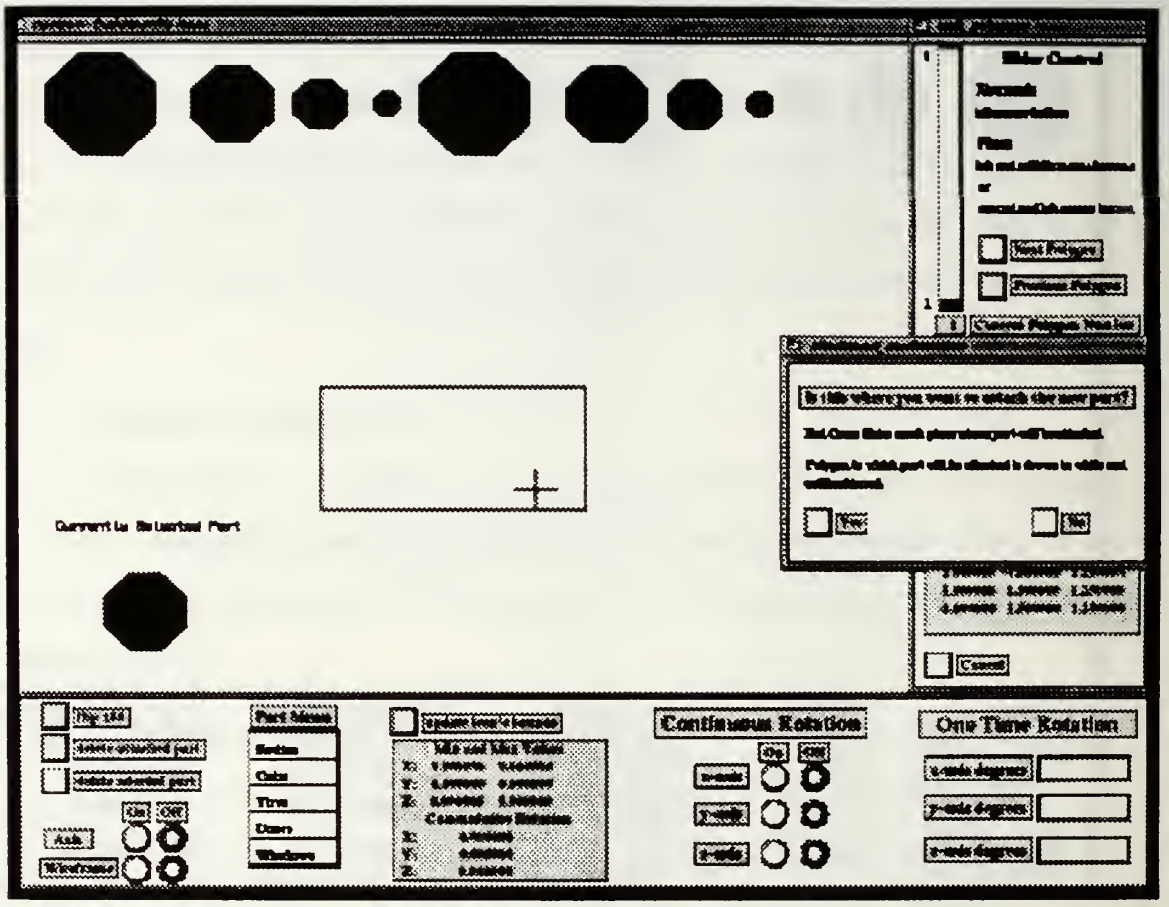

Figure A.15 - Selection of Initial Attachment Point for a New Part

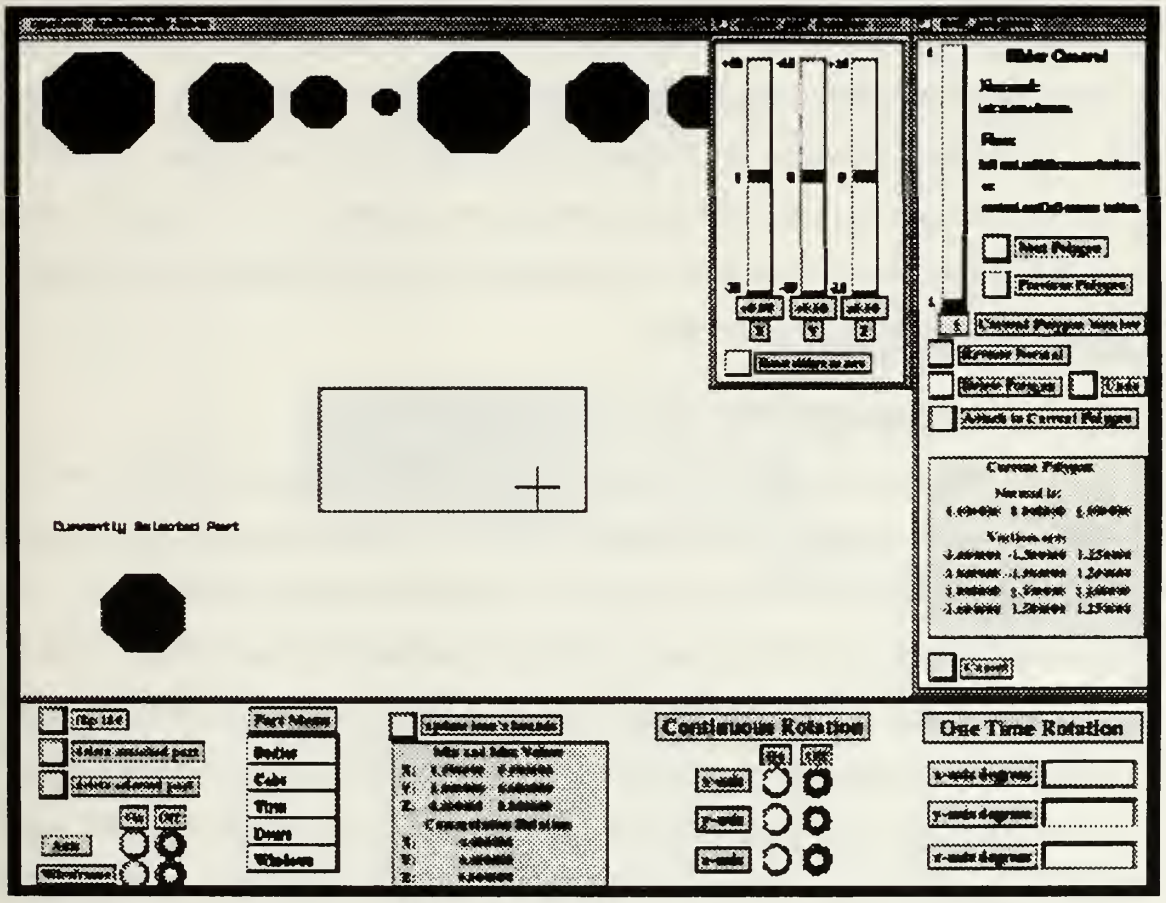

Figure A.16 - Adjustment of Attachment Point and Polygon for a New Part

the user wishes to attach his part is not highlighted in white, the user can user the slider or the next and previous polygon buttons so that it is highlighted. If the red placement cross hairs are not at the position at which the user wants his part attached, he can move them using the x, y, and z sliders in the attach part location window. When the user is happy with the polygon the part is to be attached to and the placement location, he presses the attach polygon button to attach the part to the icon. The system then puts up a window asking the user if he wants to adjust the placement of the part further (Figure A.17).

If the user selects yes, the part is attached to the icon and the user is then asked if he wants to adjust the placement of the part further (Figure A.17). If at this point the part is on backwards, the user can fix this by pressing the flip 180 degrees button found in the lower right corner of the main panel.

4. Adjustment of Part after it is Attached to the Icon

If the user decides to continue adjusting the part once he has attached it to the icon, a window is brought up to do this (Figure A.18). This window contains rotation, translation, and scaling sliders which allow the part to be rotated and scaled around the x-, y-, and z-axes and translated in the x, y, and z directions. Because the user is dealing with a three dimensional icon, he is encouraged to rotate the object to ensure the part appears properly placed from all angles. Often what looks fine from the front, doesn't look fine from a side view. When the user is satisfied with the part's placement, he signals this to the system by pressing the "Finished Adjusting Part Button" located in the lower left corner of the window to adjust the part.

5. Adding Parts Not Included in the Menu

Users can add any OFF object to the icon they are building by selecting "Load Icon" from the popup menu (Figure A.1) and then loading the file containing the desired OFF object. Initially this object will be placed in the upper left hand corner of NPSICON's main window. As this file is loaded, NPSICON brings up a window containing sliders which allow the user to rotate, translate, and scale this object (Figure A.18). NPSICON is

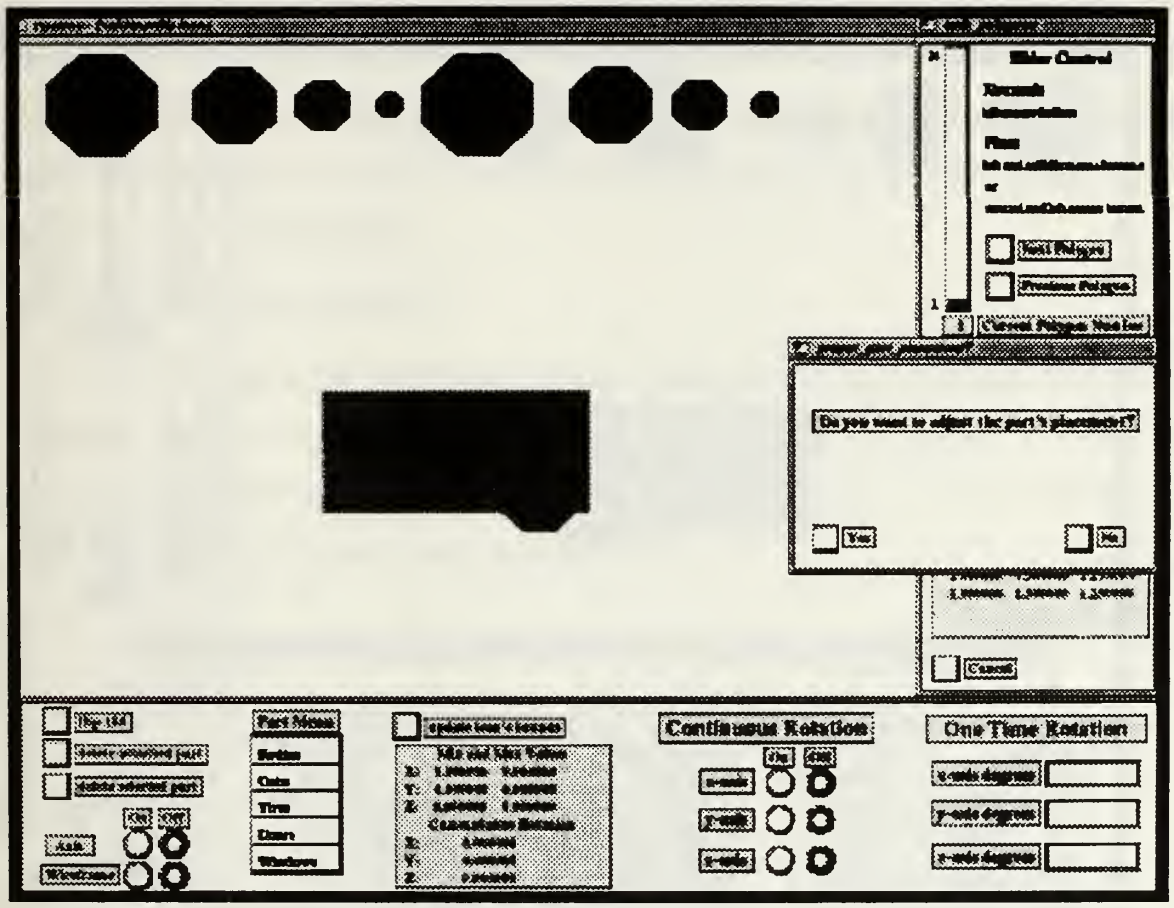

Figure A.17 - Decision on Further Adjustment to a New Part's Placement

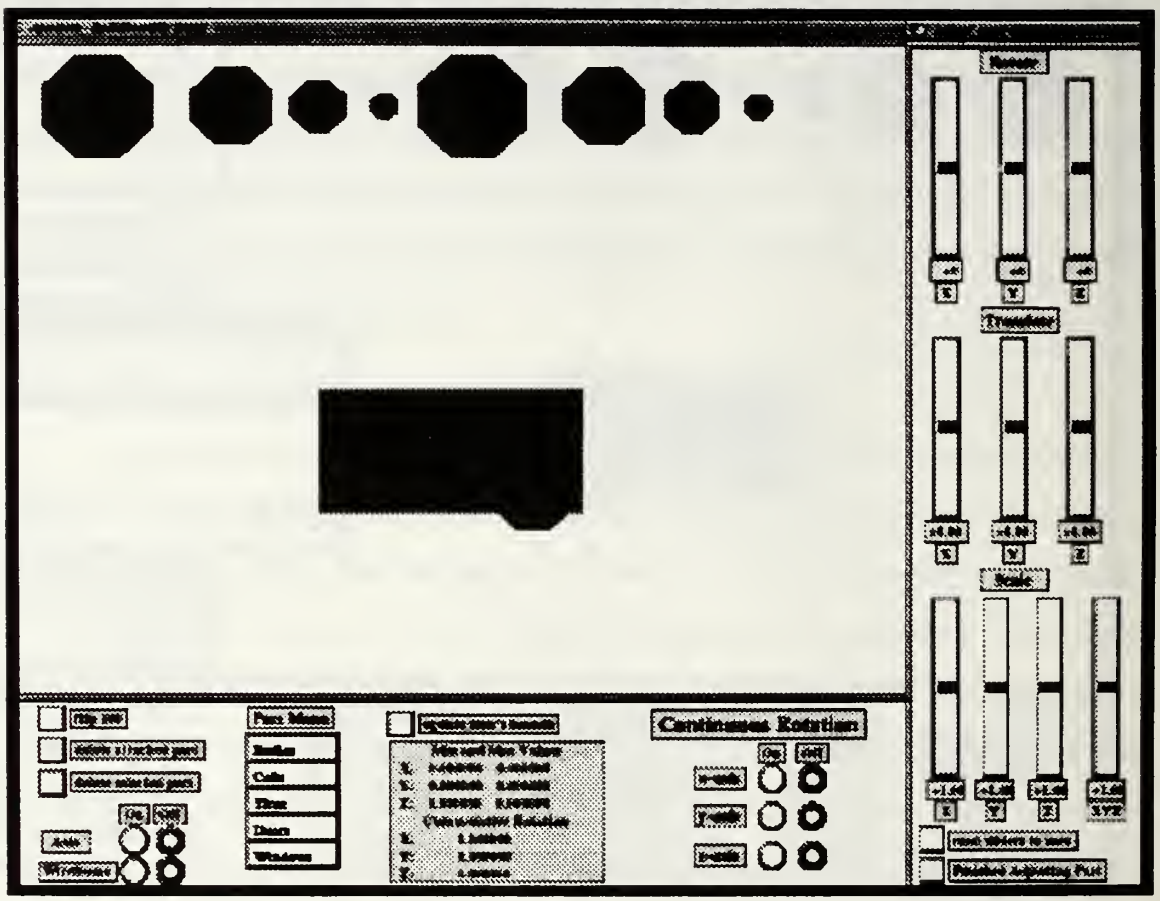

Figure A.18 - Adjustment of a New Part after Attachment to an Icon

unable to provide any automatic snapping of this object to the icon being built. Hence, theuser is totally responsible for the placement of this object with respect to the icon being built.

6. Displaying Icons Built in NPSICON

To properly display an icon built from the parts provided in NPSICON, the user must include the header file "material.h" in his display program. Material.h includes all of the definitions for the materials used within NPSICON. Material definitions are not added along with each part as the icon is being built to avoid duplication of material definitions within the icon's definition file.

M. WALKING POLYGONS

In order to look at the individual polygons making up an icon, the user selects "Walk Polygons" from the popup menu. The window which comes up contains a variety of actuators designed to help the user work with the icon's polygons as well as listing the current polygon's normal and vertices (Figure A.19).

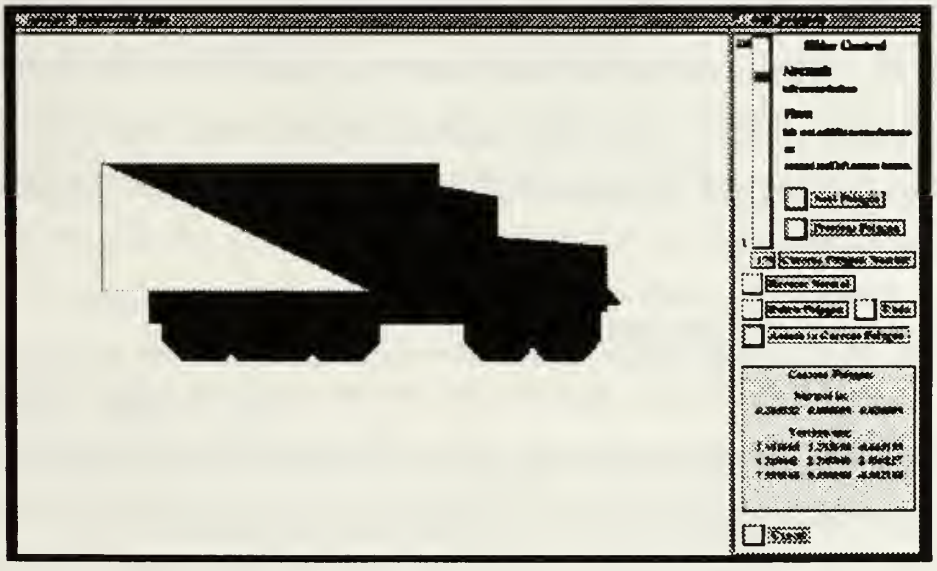

Figure A.19 - Walking an Icon's Polygons

1. Polygon Slider

The polygon slider allows the user to quickly move through all of the icon's polygons. It is operated with the left-mouse, but finer control can be obtained by holding the left and middle mouse buttons or the control key and left mouse button down simultaneously. The polygon slider displays the total number of polygons in the icon and the current polygon the user is looking at. In addition the current polygon is highlighted in white and outlined in red in the icon.

2. Next and Previous Polygon Buttons

The next and previous polygon buttons are used to allow the user to single step through the polygons in either direction. These buttons are useful when a user desires to slowly step through the icon's polygons.

3. Reverse Normal Button

The reverse normal button allows the user to reverse the current polygon's normal (i.e., to rotate it 180 degrees). In a graphics system, color is normally only applied to the front side of a polygon. A polygon's normal vector is used to tell the graphics system which side of the polygon is its front and hence which side gets the color. It is very easy to point the normal the wrong way and hence have the polygon back facing front. This is easy to detect since the polygon will appear black. The reverse normal button provides an easy way to fix this problem.

4. Delete Polygon Button

Speed is key to the success of any real-time graphical system. One way to obtain speed is to avoid doing anything unnecessary, including depicting extra or unseen polygons in an icon. Often because of the way an icon will be displayed there may be portions of it that will not be visible to the user, such as the underside of a truck. Since these polygons will not be seen, it doesn't make sense to have the graphics system display them. The delete polygon button is designed to allow the user to delete these unnecessary polygons.

5. Undo Button

The user can restore the last polygon deleted from the icon by pressing the undo button. The undo button only works on the last polygon deleted from the subicon. Undoing the deletion of one polygon does not allow a user to undo any previous deletions.

6. Attach Polygon Button

The attach polygon button is used when the user has changed the system selected polygon or changed the placement position for a new part. It signals to the system that the user is now ready to attach the selected part to the icon.

7. Cancel Button

The cancel button is used to leave the walk polygons option and return to the program.

N. CREATING A SUB-ICON

Sub-icons are icons created from the polygons of existing icons. They are useful when a user desires a copy of one part in an existing icon. To create a sub-icon, the user selects "Create Sub-Icon" from NPSICON's popup menu (Figure A.1). The user is then presented with a window which can be used to create a sub-icon by selecting polygons from the currently displayed icon (Figure A.20). The user can either delete the polygon from the icon and add it to the sub-icon, or he can add a copy of the icon's polygon to the sub-icon. After a polygon is added to the sub-icon, it is displayed in lavender. The window to walk an icon's polygons is also brought up to allow the user to locate the polygons he wishes to add to the sub-icon (Figure A.19).

Unless the user saves the sub-icon, it will cease to exist when the user exits the "create a subicon" window. Selecting the "Save Sub-Icon" button brings up the same window used to save icons. For specific details on saving files, see Part H in the Appendix. The file created when a sub-icon is saved contains no material or color for the polygons in it. To add

them, the user must exit NPSICON and using a text editor manually add them to the sub-icon's off file.

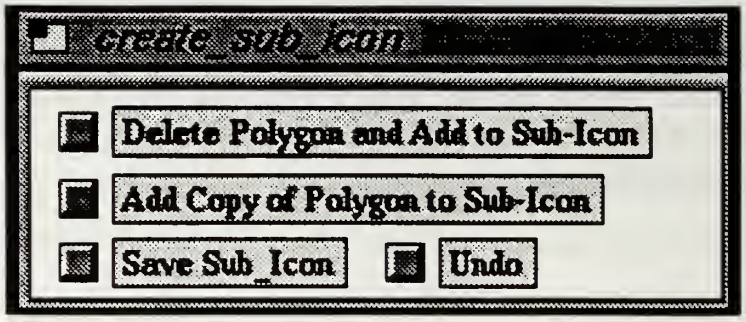

Figure A.20 - Window Used to Create a Sub-Icon

While creating a sub-icon, the user can rotate the icon and sub-icon using the continuous rotation buttons and the one-time rotation typeins located at the bottom of NPSICON's main window. To scale, rotate, and translate the sub-icon individually, the user must save the sub-icon to a file and then reload it into NPSICON using the "Load Icon" option on the popup menu (Figure A.1). He can then manipulate the sub-icon as he would any other icon in NPSICON. If the user has not added materials or colors to the sub-icon's file, the sub-icon will receive NPSICON's default material (white).

After a sub-icon is saved, the user can then create another sub-icon from the currently displayed icon. The user can repeat this process to create as many sub-icons as needed from the currently displayed icon. The user can remove the last polygon added to the subicon by pressing the "Undo" button. The "Undo" button only works on the last polygon added to the subicon. Undoing one polygon does not allow a user to undo any previous polygons. If the polygon the user wants to remove from the subicon was deleted from the icon, then it will added back to the icon when it is removed from the sub-icon. When the user has finished building and saving his sub-icon, he can exit the window to create sub-icons by pressing the "Cancel" button located in the lower left corner of the window to walk an icon's polygons (Figure A.19).

O. CLEARING DISPLAYED PARTS

To remove any parts displayed at the top of NPSICON's Main Window, the user selects "Clear Displayed Parts" from NPSICON's popup menu (Figure A.1). Users may desire to remove the displayed parts in order to better view the currently displayed icon.

P. RESETTING NPSICON

The user can return to the original start-up state of NPSICON at any time by selecting "Reset" from the popup menu. Selecting reset causes any parts, icons, or windows other than the main window to be removed from the screen so that the user will again be presented with NPSICON's initial display (see Figure A.2).

Q. EXITING THE PROGRAM

To exit the program, the user selects "Exit Program" from the popup menu and then selects yes from the window which NPSICON displays to confirm that the user really wants to exit the program.

LIST OF REFERENCES

[Akel90] Akeley, K., "The Hidden Charms of Z-Buffer," *IRIS Unvierse*, v. 11, pp. 31-37, March 1990.

[Hall89] Hall, R., *Illumination and Color in Computer Generated Imagery*, pp. 9-11, Springer-Average, 1989.

[Hanr89] Hanrahan, P., "A Survey of Ray-Surface Intersection Algorithms," in Glassner, A.S., ed., *An Introduction to Ray Tracing*, pp. 108-111, Academic Press, 1989.

[King90] King, D. M., and Prevatt, R. M., III, *Rapid Production of Graphical Interfaces*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1990.

[Meie87] Meier, T. W., *Investigation into the Use of Texturing for Real-Time Computer Animation*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1987.

[Muns89] Munson, S. A., *Integrated Support for Manipulation and Display of 3D Objects for the Command and Control Workstation of the Future*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1989.

[Nage89] Nagel, D. E., *3DShips: Rapid 3D Icon Generation for the Command and Control Workstation of the Future*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1989.

[Sili90] Silicon Graphics, Inc., *Graphics Library Programming Guide*, Version 2.0, pp. 12-1--12-12, May 1990.

[Zyda90] Zyda, M. J., *Book Number 7*, Class Notes, Naval Postgraduate School, Monterey, California, 9 January 1990.

[Zyda91] Zyda, M. J., and Pratt, D.R., "NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation," *Society for Information Display International Symposium Digest of Technical Papers*, pp. 361-364, May 1991.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943-5100	2
Dr. Michael J. Zyda Naval Postgraduate School Code CS, Department of Computer Science Monterey, CA 93943-5100	2
David R. Pratt Naval Postgraduate School Code CS, Department of Computer Science Monterey, CA 93943-5100	2
Captain Jane S. Polcrack RD #2 Box 325 Troy, PA 16947	1

Thesis
P6729 Polcrack
c.1 Using solid modeling
techniques to construct
three-dimensional icons
for a visual simulator.